Silly Semantic Circuits with SWOW

Dylan P. Gleason

***Abstract*—This paper explores the use of the Small World of Words knowledge graph to yield meaningful humorous texts by way of the methods of Labutov and Lipson who proposed a method of doing so governed by Victor Raskin's Semantic-Script Theory of Humor. These methods are employed to explore the power of different semantic networks in representing complex common senses relationships in a more generalized fashion than done so by training machine learning models. This will cut down on the need for task specific data collection and though humor is used as the illustrative example of the model's power, these methods should be employable for other complex sematic relationship tasks. The effectiveness of the model is measured through a number of different computational humor detection methods and is tested against positive and negative comparable datasets.**

I. Introduction

THE field of computational humor has, in recent years, gained some traction as NLP is becoming a much more prevalent field of study. Students in the field are beginning to use humor as a metric against which they might test the limits of modern NLP tools. With the rise of machine learning, training ML models has become the method of choice for many experimenting with the problem. However, this direction can be impractical or otherwise problematic. As with all ML models, sourcing large, tagged datasets can be difficult, as jokes often come in several different forms and structures. These datasets can also often yield problematic results, as the jokes are not filtered to be held to a moral standard and thus many of the models trained on such datasets can themselves generate harmful material. In addition to the difficulties brought on by these methods, they also do nothing to further our understanding of the mechanisms of humor. By simply parroting the most common word patterns associated with humor, ML models do little to illustrate the power of the linguistic mechanisms that yield what we consider to be humorous. For these reasons, it is in the best interest of linguists and data scientists to explore alternative methods of humor representation.

Historically, generative humor models have centered around strict rule-based generation such as wordplay, puns, riddles, etc. And although much of the linguistic academic community has generally lauded Raskin's Semantic-Script Theory of Humor (Raskin, 1985) (SSTH) since its publication in 1985 as the most apt linguistic generalization of verbal humor, it had not been adapted for the field of computational linguistics until the work of Labutov and Lipson (2012).

The Cornell researchers set out to find an existing infrastructure with which they could model a semantic script or string of related concepts. The SSTH posits that for a joke to be funny it must highlight the interplay of two semantic scripts that possess both an overlap and incongruity with one another. To capture such semantic relations the researchers opted to use the ConceptNet knowledge graph. The graph contains not only weighted relationships between concepts, but also tagged edges that use predicates to illustrate the nature of how the concepts are related (e.g., "AtLocation", "HasProperty", "UsedFor"). ConceptNet is one of the largest and most well updated knowledge graph tools available for use, however the purpose of this paper is to explore the power of different knowledge graphs beginning with a much less used knowledge graph called the Small World of Words.

.

II. SWOW

Small World of Words (De Deyne, 2019) (SWOW) knowledge graph is a project funded by the Flemish Research Council and through the University of Leuven Research Council, Belgium. SWOW has become an international project now containing cues and responses from 17 languages, though for the purposes of this paper only the English dataset will be used. It has been collecting natural language data from 2011 through a simple word association game. Participants are given a cue word and are asked to respond to that cue with the first 3 words that come to mind. Because of the open-ended nature of the data collection, there are inconsistencies in the data, but the system is much simpler than ConceptNet and other such knowledge graphs, and still is able to yield comparable computational power.

A 2021 paper from students at the University of Melbourne (Liu, 2021) was able to compare SWOW directly with ConceptNet on being able to yield strings of concepts that represent common sense knowledge that can be computationally actionable. This is despite the obvious disparity in resources between the two. The study found that although SWOW is significantly smaller than ConceptNet

Table 1. Example of SWOW Data Entries

Participant ID	Age	Gender	Native Language	Country	Cue	R1	R2	R3
32656	31	Ma	United States English	United States	Embrace	Hold	Grab	Hug
32656	31	Ma	United States English	United States	Conversation	Talk	Speak	Listen
32656	31	Ma	United States English	United States	Ouch	Hurt	Pain	Farm
32656	31	Ma	United States English	United States	Fault	Car	Insurance	Traffic

Figure 1 Each participant has to provide 3 responses to the cue word. As seen above some of the responses are nonsensical, but the multiple answers allow participants to tie the cue to, potentially, semantically unrelated concepts to make the data more robust.

concept, because SWOW does not provide information on nature of relationship, second concepts were chosen as one of the responses to the cue concept, and the following chained concept would be chosen as a cue to which concept B would be a response. This back and forth of cue to response would help categorize the nature and POS of each concept generally, to ensure that it would yield a semantically meaningful connection to its surrounding nodes. Chains could continue in this manner for as long as researchers see fit in these initial stages, however complications would arise relating to script length in later stages. Each node is finally ranked by its conditional probability relating to the given edge weight of the preceding and succeeding nodes, and using each node weight an overall weight is assigned to the script to help with ranking. Using this method, it was observed that script weights higher than 5.5e-07 typically yielded scripts with a more cohesive narrative although this measure of ranking was not used in later stages.

III. Script Relation Methods

Once a model of generating single scripts was working, thus began the larger task of quantifying script relations between each other. As stated before, for a piece of text too be humorous according to SSTH the setup must illustrate the semantic overlap of the two scripts stemming from a common seed and then the punchline must realize a glaring incongruity between the two scripts. So, for these scripts to yield humorous texts they must have both overlap and incongruity.

To realize an amount of inter-script overlap, the Labutov and Lipson study utilizes a fan-out spreading activation. To generate a weight for a given node, the sum of all (pre-normalized) conditional probabilities relating to all potential scripts stemming from the current node is taken. The original paper calls for a decay factor to penalize long scripts but this study went the route of keeping script length static and short across the board to deal with large runtime issues. Once the weights for the seed concept node, and all potential B level and C level nodes are calculated, the lists for B and C level nodes are filtered to only include nodes that have above average node weight. Then finally, once nodes are calculated each potential script is ranked as a sum of the log of each of the node weights. Each of the highest-ranking pairs of scripts consisting of only the highest-ranking individual nodes represent scripts with

quantified, the scripts then must be vetted for their incongruity. In order to represent this, we cluster the nodes of the entire SWOW graph using the Fast Community Detection Algorithm. Each community yielded by the algorithm represents a very rough semantic category containing words that all either come in a similar grammatical form or relate to a common theme. Using these communities, nodes in the existing list of valid script candidates are tagged in one of the communities. During natural language realization, the concepts used in the setup portion of the template all belong to the same community as the seed concept to ensure close overlap twofold. Then the concept introduced in the punchline portion of the template is specifically the outlier concept from the semantic circuit that belongs to a different community. The setup portion is in the form of a question that relates the seed concept to one other related concept from the two scripts, and the punchline is in the form of an answer that relates the seed concept to an incongruous concept still belonging to the same semantic circuit. The script candidates are all tagged for POS and those tags are used to place them into the appropriate template that will realize them with grammatical consistency. The next section will provide a more detailed step-by-step description of the algorithm used.

IV. Algorithm

This section details each step of the humor generation process based on the Labutov and Lipson methods for ease of recreation and experimentation.

A. *Import Libraries:* The code starts by importing necessary libraries such as pandas, NumPy, random, NLTK, and networkx.

B. Read Data: The SWOW CSV file containing information about concepts and their relationships strengths is read into a DataFrame named df_strength.

C. *Select a Concept:* A random number is generated to choose a seed concept from the DataFrame, and the corresponding concept is printed.

D. *Generate Data for Seed Concept:* A subset of the DataFrame is created to include all rows where the 'cue' column matches the selected seed concept.

E. *Define Function for Chaining Concepts:* In this step, the chain_concept function is defined to generate a concept based on probabilities and strength values. Here's a breakdown of its key components:

- *Input:* The function takes the seed concept as input.
- *Probability Calculation:* It calculates a distribution number, multiplies it with the strength values associated with responses to the key concept, and shuffles the probabilities.
- *Threshold Calculation*: It sets a threshold based on the distribution number.
- *Cumulative Sum:* It iterates through the shuffled probabilities, calculates the cumulative sum, and breaks the loop when the cumulative sum exceeds the threshold.
- *Output:* The function returns the selected concept and the probability associated with it.

F. *Calculate Long-Range Probabilities:* In this step, the code calculates the total probability of a chain of concepts starting from the seed concept and going through multiple steps. Here's how it works:

- *Activation Probability:* The probability of the seed concept (Prob_of_A) is calculated as the ratio of the number of occurrences of the seed concept to the total number of concepts in the dataset.
- *Loop Through Concepts:* For each concept (B) related to the seed concept, the code calculates the conditional probability of reaching B from the seed concept. This involves multiplying the probability of B given A by the probability of A to B.
- *Cumulative Total Probability A:* The total probability of reaching any concept starting from the seed concept is the sum of all conditional probabilities.

G. *Sort and Filter Probabilities*: In this step, the code sorts and filters probabilities based on certain conditions. Here's a detailed breakdown:

- *Sorting Probabilities for Concept B*: The code iterates through all possible responses (B) to the seed concept and calculates the total probability of reaching B from the seed. It sorts these probabilities in descending order.
- *Finding a Threshold:* The total probabilities are summed, and a threshold is set as the average of these probabilities.
- *Filtering Based on Threshold:* The probabilities that exceed the threshold are considered significant, and the corresponding concepts are filtered into a list.
- *Repeat for C:* The process is repeated one final time to calculate the probabilities at every possible C level node

H. *Community Detection:* In this step, the Louvain algorithm is applied to detect communities within the network of concepts. The Louvain algorithm is a widely used method for community detection in network analysis.

- *Convert DataFrame to Graph:* Convert the DataFrame containing relationships between concepts into a graph representation. In this graph, concepts are nodes, and relationships between them are edges.
- *Apply Louvain Algorithm:* Use the Louvain algorithm to identify communities within the graph. The algorithm aims to maximize the modularity of the network, effectively partitioning it into groups of densely connected nodes.
- *Extract Communities:* Extract the communities based on the partition information. Each node (concept) is assigned to a specific community.

I. *Generate Jokes Based on Templates:* Extract the communities based on the partition information. Each node (concept) is assigned to a specific community.

- *Define Joke Templates:* Simplistic joke templates are created to accommodate words based on their part-of-speech tag to realize natural sounding sentences.
- *Iterate Through Script Pairs:* Iterate through pairs of scripts, each representing a combination of concepts from different communities.
- *Separate Concepts with Same Community Tag*: Separate the concepts into those that have the same community tag as the seed and those that don't.
- *Generate Jokes Using Templates*: Selecting the template that best corresponds to the parts-of-speech of each concept, the seed concept and concepts in the seed concept community are placed in the setup part of the joke, and the concepts with other community tags are put in the punchline.

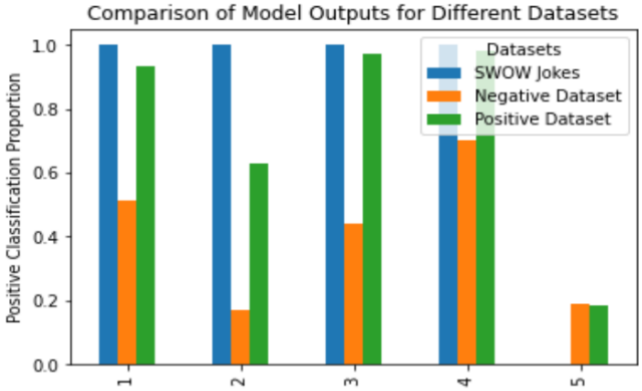

Figure 2 All AI detection models classify the SWOW generated jokes as humorous

V. Results

These methods did indeed end up synthesizing some feasible humorous Question/Answer texts as designed. Although testing for humor in a text can be a bit of a subjective practice, we do have more tools to do so now than we did in the past. The original paper used human feedback to rate the level of humor against other positive and negative text examples to some luck. Although this study did not have the means to crowdsource opinions, we were able to use computational means to measure the humor in these texts. Using 5 different ML models of humor detection (Baladon, 2023; Dhiab, 2023; r3b3lj3l, 2023; Reggie, 2023; Wan, 2022) the corpus of SWOW synthesized jokes were able to achieve very positive results, generating 100% positive predictions on 4 out of the 5 models. The models were also tested against a human written Questions/Answer Joke corpus (Roznovjak, 2017) as a positive baseline and human written Questions/Answer non-humor corpus (Allurophile, 2020) as a negative baseline, both of equal size to the SWOW generated corpus. The 100% result may be troubling however, due to the fact that time constraints and high computational complexity on generating C level node probability weights to factor towards the overall scripts weights restricted this study to only generating a small corpus of 1000 example on the same seed concept. The homogeneity of the corpus likely has much to do with the outcome, as the 4 models that yielded positive results did so at 100% and the 1 model that yielded negative results did so at 100%. Further work generating a much larger and more diverse dataset will be needed to draw more meaningful insights from these methods, though the preliminary results are promising.

VI. Future Work

As mentioned above issues concerning the large runtime and subsequent sub-optimal data are the first to address to gain any more insight from this study. Once a reasonable optimization of the node overlap weighting process is achieved, larger datasets could be generated as well as longer scripts that would more closely align with the methods of the original study, namely the use of decay weighting to generate longer and more organic narrative scripts, which would then allow for a more traditional semantic circuit. The punchline concept being a member of both scripts and from a separate community should present a starker contrast between overlap and incongruity which should produce more humorous texts. Future work should also include comparisons against other knowledge graph representations to determine what methods of knowledge graph curation can represent the most abstract semantic relationships. To further illustrate the point of these methods being used as a generalized alternative to ML humor generation, work should be conducted comparing the two methods more side by side. Further work should also be put towards the end of making a more distinct quantification method of determining when a text is humorous or not so that these methods may be more properly rated.

Acknowledgment

A huge thank you to both Dr. Srinivas Bangalore and Dr. and Feldman for your continued support and limitless patience. You can find the source code for this project on GitHub at https://github.com/dylangleason5/SWOW_Humor.

References

Allurophile. (2020). Question-Answer combination [Dataset]. Retrieved from https://www.kaggle.com/datasets/veeralakrishna/questionanswer-combination

Baladon, Alexis (2023). *autotrain-huhu-humor-54189127188*. Retrieved from https://huggingface.co/transformers

Dhiab, Mohamed (2023). */humor-no-humor*. Retrieved from https://huggingface.co/transformers

Labutov, G. I., & Lipson, H. (2012). Humor as Circuits in Semantic Networks [Review of Humor as Circuits in Semantic Networks]. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 150–155.

De Deyne, S., Navarro, D. J., Perfors, A., Brysbaert, M., & Storms, G. (2019). The "Small World of Words" English word association norms for over 12,000 cue words. [Review of The "Small World of Words" English word association norms for over 12,000 cue words]. *Behavior Research Methods*, *51*, 987–1006. https://doi.org/10.3758/s13428-018-1115-7

Liu, C., Cohn, T., & Frermann, L. (2021). Commonsense Knowledge in Word Associations and ConceptNet [Review of Commonsense Knowledge in Word Associations and ConceptNet]. *Proceedings of the 25th Conference on Computational Natural Language Learning (CoNLL)*, 481–495. Association for Computational Linguistics.

r3b3lj3l (2023). *humor_classifier*. Retrieved from https://huggingface.co/transformers

Raskin, V. (1985). Semantic mechanism of humor. D. Reidel Publishing Company.

Reggie (2023). *distilbert-joke_detector*. Retrieved from https://huggingface.co/transformers

Roznovjak J. (2017). Question-Answer Jokes [Dataset]. Retrieved from https://www.kaggle.com/datasets/jiriroz/qa-jokes

Wan, Thana (2022). *bert-base-uncased-finetuned-humordetection*. Retrieved from https://huggingface.co/transformers